Carletti Luca cavejo
Lorenzin Massimo lorenz
Antoniazzi Stefano antoniaz
Maccà Nicola macca
Parise Fabio parise
Pietribiasi Nicola pietri

CLAMPP group

Business Card Exchange

User Manual

Index

Tutorial section

Section 1 : Introduction..... 3

About this tutorial..... 3
What the program does..... 3
Requirements..... 3
Preliminary operations (WINDOWS 3.1, WINDOWS95, WINDOWSNT)..... 3

Section 2 : Writing and storing simple Business Cards..... 4

Example of a simple Business Card..... 4
How to store a Business Card written on a text file..... 4
How to store a Business Card arrived via e-mail..... 4
How to remove a Business Card..... 5

Section 3 : Writing and using simple .FOR file..... 6

Introduction..... 6
Example of a simple .FOR..... 6
How to use a .FOR..... 7

Section 4 : Update your HTML page with Business Card data.... 8

Introduction..... 8
Example of a simple HTML file with Business Card fields..... 8

Section 5 : Who wrote us a Business Card ?..... 10

Getting a list on video..... 10
Getting a list on a text file..... 10
Getting all the information about a person on video..... 10
Getting all the information about a person on a text file..... 10

Section 6 : Examples of Business Cards..... 11

Introduction..... 11
Examples..... 11

Section 7 : Examples of .FOR files..... 15

Introduction..... 15
Examples..... 15

Reference section

Section 8 : Table of Business Card fields..... 20

Section 9 : Schemes of .FOR files rules..... 21

Section 10 : More about HTML pages..... 27

Section 11 : The alias file "ALIAS.BCA"..... 28

Section

1

Introduction

About this tutorial

This tutorial is for the DOS based command line version. A tutorial for the windows version is available on the online help.

What the program does

The program reads text files containing personal information written in a particular format (we will call it Business Card format) and produces text files that can be imported by a PIM (Personal Information Manager).

If Mail files have personal information written in Business Card format, the program will also store those personal information.

Requirements

The program BCEX.EXE is the command line version of the program. It can be used by a stand alone program on all the DOS based systems.

There are 2 graphical interface versions of the program :

1. 16 bit version for WINDOWS 3.1

2. 32 bit version for WINDOWS 95 and WINDOWS NT

Preliminary operations WINDOWS 3.1, WINDOWS 95 and WINDOWS NT

Take care of having the right version for your system (32 bit version will not run on WINDOWS 3.1)

Run SETUP.EXE from drive A :

The installation program will ask you the directory where to put the program, if you want to create a program group and the icon for the program.

Section

2

Writing and storing simple Business Cards

Example of a simple Business Card

Edit this file with the standard DOS text file editor, taking care of being in the same directory of the program :

1. Type at the DOS prompt : edit example.txt
2. Type the following text and save the file

<business card>

<name> John **</name>**

<surname> Smith **</surname>**

<email> john@website.com **</email>**

</business card>

How to store a Business Card written on a text file

Now you can store the file example.txt in the program with the command :

BCEX -readtext text=example.txt

Attention :if you edit the file example.txt being on a different **directory** (for example c :\user) you have to write the command BCEX -readtext text=c :\user\example.txt

How to store a Business Card arrived via e-mail

Send yourself an email containing (this means that you may write the following text where you want in the email) the Business Card :

<business card>

<name> Rose **</name>**

<surname> Madder **</surname>**

<email> rose@website.com **</email>**

<home phone> 11122333 **</home phone>**

</business card>

At this point you have to know where your mail manager put the mail files. Consider that the email are stored in the file C :\Mailman\mailbox.txt

Now you can store the Business Card arrived with the command

BCEX -readmail mail=C :\Mailman\mailbox.txt

When you'll have many Business Cards arrived via e-mail to manage you may specify what Business Cards you want to store specifying a date. Only the Business Cards arrived after that date will be stored.

Consider that you want to store only the Business Cards arrived after the date January 30th, 1997. You have to use the command :

BCEX -readmail mail=C :\Mailman\mailbox.txt date=30/01/1997

Attention : as you can see the **date** is specified in **European** format. The year must be written in **4 figures** format.

How to remove a Business Card

For removing a Business Card you have to specify name, surname and e-mail. Suppose you stored the Business Card of Mr. Smith as seen at page 4 (name=John, surname=Smith, email= john@website.com). For removing it you have to use the command :

BCEX -remove id=John-Smith-john@website.com

Section

3

Writing a simple .FOR file

Introduction

PIM means Personal Information Manager. Usually PIMs have the possibility to import data from text files. Unfortunately there isn't a standard for these text files, even if a lot of PIMs support the CSV (Comma Separated Values) format. So the Business Card Exchange program have a text file (we call it .FOR file) in which is possible to specify what kind of text file the PIM needs.

Example of a simple .FOR file

Suppose to have a PIM that imports data from a text file. Suppose that the text file must be written in such a way :

1. every field (in our program a field concerns personal information so a field is **name**, another field is **surname** and so on) must be separated by a comma.
2. no other characters are needed to wrap text fields.
3. for separating data related to different persons is used a line feed.

An example of a text file that can be imported by such a PIM could be :
John,Smith,john@web.com,USA
Fabio,Parise,fabio@maya.it,ITALY,
Rose,Madder,rose@ger.com,GERMANY

Our problem now is to produce such a file having stored the Business Cards :

John Business Card	**Fabio** Business Card	**Rose** Business Card
<business card>	<business card>	<business card>
<name> John </name>	<name> Fabio </name>	<name> Rose </name>
<surname> Smith </surname>	<surname> Parise </surname>	<surname> Madder </surname>
<email> john@web.com </email>	<email> fabio@maya.it </email>	<email> rose@ger.com </email>
<home country> USA </home country>	<home country> ITALY </home country>	<home country> GERMANY </home country>

</business card>	</business card>	</business card>

Edit this file with the standard DOS text file editor, taking care of being in the same directory of the program :

1. Type at the DOS prompt : edit example.for

2. Type the following text and save the file

<DATA><name>,<surname>,<email>,<home country>**</DATA>**

How to use a .FOR

At this point you can produce the text file (for example I'll call it forpim.txt) that will be imported by the PIM with the command :

BCEX -export for=example.for fileout=forpim.txt

Now is possible to use the PIM import function with the file the file forpim.txt.

Attention : if the file example.for is in the **directory** c :\myfor and you want to produce the file forpim in the directory c :\mypim you have to use the command :BCEX -export for=c :\myfor\example.for fileout=c :\mypim\forpim.txt

Section

4

Update your HTML page with Business Card data

Introduction

An HTML page is a text file with particular fields that allow the browser to do something. These fields identifiers are called tags. For example what must be shown on the screen by the browser is identified by the beginning tag <body> and the end tag </body>. We will use the same tag technique for letting the Business Card Exchange program to identify what kind of information it's going to manipulate.

Example of a simple HTML file with Business Card fields

Edit this file with the standard DOS text file editor, taking care of being in the same directory of the program :

1. Type at the DOS prompt : edit example.htm

2. Type the following text and save the file

```
<HTML>
  <HEAD>
    <TITLE> Business Card FORM </TITLE>
  </HEAD>
  <BODY>
    I am <name>John</name> <surname>Smith</surname> and I work at
    <work company>Raw INC.</work company>
  </BODY>
</HTML>
```

Now you can open your HTML browser and watch what appears : there will not be any trace on the screen of the Business Cards fields here written in bold style.

You can change this HTML page with new data. You must edit a text file with a single Business Card (see page 5 : writing a simple Business Card).

For example edit this file with the standard DOS text file editor, taking care of being in the same directory of the program :

1. Type at the DOS prompt : edit bcsample.txt

2. Type the following text and save the file

<business card>

<name> John **</name>**

<surname> Smith **</surname>**

<email> john@website.com **</email>**

<work company> FinCom INC. **</work company>**

</business card>

Now you can use the command :

BCEX -html htmlfile=example.htm bcfile=bcsample.txt

Attention : if the file bcsample.txt is in the **directory** c :\mybc and you want to produce the html file example.htm in the directory c :\myhtml you have to use the command :
BCEX -html htmlfile=c :\myhtml\example.htm bcfile=c :\mybc\bcsample.txt

The HTML file example.htm updated will be :

```
<HTML>
   <HEAD>
      <TITLE> Business Card FORM </TITLE>
   </HEAD>
   <BODY>
      I am <name>John</name> <surname>Smith</surname> and I work at
      <work company>FinCom INC.</work company>
   </BODY>
</HTML>
```

Section

5

Who wrote us a Business Card ?

Getting a list on video

For having the surname, name and email information about who wrote us a Business Card (note that these Business Cards must be stored with the command specified at page 3) there is the command

BCEX -getlist

Getting a list on a text file

For having on a text file "list.txt" the surname, name and email information about who wrote us a Business Card (note that these Business Cards must be stored with the command specified at page 3) there is the command

BCEX -getlist fileout=list.txt

Getting all the information about a person on video

For having on video all the information about a person identified by name, surname and email (for example John Smith john@net.com) the command line function is

BCEX -view id=John-Smith-john@net.com

Getting all the information about a person on a text file

For having on a text file "person.txt" all the information about a person identified by name, surname and email (for example John Smith john@net.com) the command line function is

BCEX -view id=John-Smith-john@net.com fileout=person.txt

Section

6

Examples of Business Cards

Introduction

A Business Card may be sent via e-mail (signature files may help in doing it automatically) or may be stored in a text file.

There are several fields you can insert in your Business Card, but first of all it is essential to define the start and the end of the card itself. Every Business Card has to start with the begin tag **<Business Card>** and has to finish with the relative end tag **</Business Card>**. Controls are not case-sensitive, so you can write every tag using capital letters or not, but it's important not to introduce other blank characters between the single words.

To identify one person, you have to insert three essential information: the surname, the name and the email address. To do this you have to write them into a tag sequence using standard tags **<surname>**, **<name>** and **<email>**

Attention : each Business Card is well defined after specifing these 3 fields **name**, **surname**, **email**. If only one is omitted, the entire Business Card will be rejected, so be careful.

Every value that you insert belongs to a certain category. The value inserted must be between its related beginning tag and ending tag. For example a name must be inserted between <name> and </name> and the result may be for example :

<name> Deborah **</name>**

Examples

Look at page 4 for step by step instructions. Substitute the simple Business Card with the following Business Cards.

(1)

<business card>
<surname> Kirk **</surname>**
<name> James **</name>**
<email> james@Enterprise.StarTrek **</email>**
</business card>

Each business card is well defined after specifing the 3 fields name, surname and email. If only one is omitted, the entire business card will be rejected, so be careful.

(2)

You can also indicate more than one email address or web pages using repeatedly the relative tags <email> and <url>.

<business card>
<surname> Kirk **</surname>**
<name> James **</name>**
<email> james@Enterprise.Star.Trek **</email>**
<email> james@Genesis.Planet **</email>**
<url> www.Enterprise.Capitain **</url>**
</business card>

The url and email tags are the only one you can duplicate. In the other cases only the first occurrence will be accepted.

(3)

Another useful field is the date. User can write his date with the syntax dd/mm/yyyy. If your BC doesn't contain a date field, the program will introduce it reading the current date of the system.

<business card>
<date> 02/12/2399 **</date>**
<surname> Kirk **</surname>**
<name> James **</name>**
<email> james@Enterprise.StarTrek **</email>**
<email> james@Genesis.Planet **</email>**
<url> www.Enterprise.Capitain **</url>**
</business card>

(4)

Many other fields allows you to include several information like home address, work address, home phone, etc... All these ones have to be written in the same tagged form discussed before.
However, you could need some extra-fields not included in the standard list. For this reason there is a special keyword "custom:" that allows you to create as many tags as you want. This is the right use:

<business card>
<date> 02/12/2399 **</date>**
<surname> Kirk **</surname>**
<name> James **</name>**
<salutation> Admiral **</salutation>**
<email> james@Enterprise.StarTrek **</email>**
<email> james@Genesis.Planet **</email>**
<url> www.Enterprise.Capitain **</url>**

<custom:login> james **</custom:login>**
<custom:Planet> Earth **</custom:Planet>**
<custom:transmitter code> A22 **</custom:transmitter code>**

</business card>

As separator characters outside the tags, you may also use blanks, tabulation and carriage return.

(5)

Symbols like “<” and “>” are very particular: infact they rappresent the start and the end of a tag. For this reason they can’t be introduced between two tags.
For example **:**

<name> <James> **</name>** is **wrong !**)

The program tries to find a match for the tag <James> but this is not a valid tag, so the entire field will be rejected. The only way to insert these special characters into a field, consists of writing them with the prefix char “\” (back slash).
So the correct form of the last line is :

<name> \<James\> **</name>**

Now the way to include the back slash as a single character is to write it two times (\\ ) for example:

<name> < John \ Jake \ Paul > **</name>** is **wrong !**

has to be changed to:

<name> \< John \\ Jake \\ Paul \> **</name>**

A correct complete example is :

<business card>
<surname> Kirk **</surname>**
<name> James **</name>**
<email> james@Enterprise.Star.Trek **</email>**
<custom :friends>\< John \\ Jake \\ Paul \> **</custom :friends>**
</business card>

Section
7

Examples of .FOR files

Introduction

A .FOR file is a text file in which are stored information about how to produce the text file that will be imported by a PIM. We have to distinguish what must be written for every field (usually a separator like a comma), what must be written just once in the file (usually an header) and what must be not written if the field content is not present.

As general rule we will say that :

1. what lies between <text> and </text> will be written once in a file.
2. what lies between <data> and </data> will be written once for person.
3. what lies between <ifany> <"tag"> </ifany> will be written only if the field content of "tag" is present in the Business Card.
4. inside <data> and </data> is possible to write Business Cards tags in a particular format. The program will substitute the corresponding values for every person.

Examples

Look at page 6 for step by step instructions. Substitute the simple .FOR files with the following .FOR files and repeat the same operations for producing different output.

(1)

Save a text file calling it example.for containing this line :
<data><name>,<surname>,<email/0/,>**</data>**

Here we have the data field email that may contain multiple values.
Writing <email/0/,> we specify that in that position we want all the email of that person (0 is a kind of special symbol for meaning all the values) separated by a comma.

If the Business Cards stored are :

Harris card	**Smith** card
<business card>	<business card>
<name> James </name>	<name> John </name>
<surname> Harris </surname>	<surname> Smith </surname>
<email> harris@web </email>	<email> john@web </email>
	<email> john@www </email>
</business card>	</business card>

After executing the program with this command line :
BCEX -export for=example.for fileout=out.txt

The file out.txt will be :
James,Harris, harris@web
John,Smith,john@web,john@www

(2)

Save a text file calling it example.for containing this line :
<data><name>,<surname>,<email/3/,>**</data>**

Here we have the data field email that may contain multiple values. Writing <email/3/,> we specify that in that position we want at the most 3 email of that person (specifying a number n>1 will always force the program to use n-1 separators) separated by a comma.

If the Business Cards stored are :

Harris card	**Smith** card
<business card>	<business card>
<name> James </name>	<name> John </name>
<surname> Harris </surname>	<surname> Smith </surname>
<email> harris@web </email>	<email> john@web </email>
	<email> john@www </email>
</business card>	</business card>

After executing the program with this command line :
BCEX -export for=example.for fileout=out.txt

The file out.txt will be :
James,Harris, harris@web,,
John,Smith,john@web,john@www,

(3)

Save a text file calling it example.for containing this line :
<data><name>,<surname>,<email/,>**</data>**

Here we have the data field email that may contain multiple values. Writing <email/,> we don't specify how many email of that person we want so the program will use the default number (=1) (when we don't specify how many elements of that field we want or if we write 1 the separator will be printed only if there's at least a

corresponding value).

If the Business Cards stored are :

Harris card	**Smith** card
<business card>	<business card>
<name> James </name>	<name> John </name>
<surname> Harris </surname>	<surname> Smith </surname>
	<email> john@web </email>
	<email> john@www </email>
</business card>	</business card>

After executing the program with this command line :
BCEX -export for=example.for fileout=out.txt

The file out.txt will be :
James,Harris
John,Smith,john@web,

(4)

Save a text file calling it example.for containing this line :
<text> This is an header **</text>**
<data><name>,<surname>,<email/0/,>**</data>**
<text> This is a tail **</text>**

Here we have 2 text fields. In both situation the program will write only once in the file what is written inside.

If the Business Cards stored are :

Harris card	**Smith** card
<business card>	<business card>
<name> James </name>	<name> John </name>
<surname> Harris </surname>	<surname> Smith </surname>
<email> harris@web </email>	<email> john@web </email>
	<email> john@www </email>
</business card>	</business card>

After executing the program with this command line :
BCEX -export for=example.for fileout=out.txt

The file out.txt will be :
This is an header
James,Harris, harris@web
John,Smith,john@web,john@www
This is a tail

(5)

Save a text file calling it example.for containing this line :

<data><name>,<surname>,**<ifany>** maybe <email> **</ifany></data>**

Here we have the ifany field. What's inside this field is written only if there's at least one value associated to the specified Business Card tags contained.

If the Business Cards stored are :

Harris card	**Smith** card
<business card>	<business card>
<name> James </name>	<name> John </name>
<surname> Harris </surname>	<surname> Smith </surname>
	<email> john@web </email>
	<email> john@www </email>
</business card>	</business card>

After executing the program with this command line :
BCEX -export for=example.for fileout=out.txt

The file out.txt will be :
James,Harris
John,Smith,maybe john@web

This is important for those PIM that uses separators only if a value is present.

(6)

Save a text file calling it example.for containing this line :
<data>
<name>,<surname>,<email/0/,>
</data>

Here we have the data field containing some line feed. The program will write these line feed for every person.

If the Business Cards stored are :

Harris card	**Smith** card
<business card>	<business card>
<name> James </name>	<name> John </name>
<surname> Harris </surname>	<surname> Smith </surname>
<email> harris@web </email>	<email> john@web </email>
	<email> john@www </email>
</business card>	</business card>

After executing the program with this command line :
BCEX -export for=example.for fileout=out.txt

The file out.txt will be :

James,Harris, harris@web

John,Smith,john@web,john@www

(7)

Save a text file calling it example.for containing this line :
<data><name>,<surname>,<email/3/$32>**</data>**

Here we have the data field email in which is specified the separator using its ASCII decimal code (decimal number 32 is for comma).

If the Business Cards stored are :

Harris card	**Smith** card
<business card>	<business card>
<name> James </name>	<name> John </name>
<surname> Harris </surname>	<surname> Smith </surname>
<email> harris@web </email>	<email> john@web </email>
	<email> john@www </email>
</business card>	</business card>

After executing the program with this command line :
BCEX -export for=example.for fileout=out.txt

The file out.txt will be :
James,Harris, harris@web,,
John,Smith,john@web,john@www,

(8)

Save a text file calling it example.for containing this line :
<data>\<<surname>\>,\<<name>\>,\<<email/\>><**data>**

Here we have the data field containing special characters ('<' and '>' ; but also '/' and '\' and '$' are special characters too) so we must write each of them preceded by the character '\'.

If the Business Cards stored are :

Harris card	**Smith** card
<business card>	<business card>
<name> James </name>	<name> John </name>
<surname> Harris </surname>	<surname> Smith </surname>
<email> harris@web </email>	<email> john@web </email>
	<email> john@www </email>
</business card>	</business card>

After executing the program with this command line :

BCEX -export for=example.for fileout=out.txt

The file out.txt will be :
<James>,<Harris,<harris@web>
<John>,<Smith>,<,john@web>

(9)

Save a text file calling it example.for containing this line :
<data><name>,<surname>,<email/0/,& !>**</data>**

Writing <email/0/,& !> we specify that we want all the email of that person (0 is a kind of special symbol for meaning all the values) separated by the symbols ,& !.

If the Business Cards stored are :

Harris card	**Smith** card
<business card>	<business card>
<name> James </name>	<name> John </name>
<surname> Harris </surname>	<surname> Smith </surname>
<email> harris@web </email>	<email> john@web </email>
	<email> john@www </email>
</business card>	</business card>

After executing the program with this command line :
BCEX -export for=example.for fileout=out.txt

The file out.txt will be :
James,Harris, harris@web
John,Smith,john@web,& !john@www

This example shows that multiple separators are allowed.

Section

8

Table of Business Card fields

Begin Tag	Value Example	End Tag
<name>	John	</name>
<surname>	Smith	</surname>
<salutation>	mister	</salutation>
<url>	www.dei.unipd.it/home/smith	</url>
<email>	smith@dei.unipd.it	</email>
<mobile phone number>	7514856	</mobile phone number>
<mobile phone prefix>	0337	</mobile phone prefix>
<home address>	via Michelangelo 31	</home address>
<home zip>	36051	</home zip>
<home city>	Creazzo	</home city>
<home state>	Vicenza	</home state>
<home country>	Italia	</home country>
<home country prefix>	++39	</home country prefix>
<home phone prefix>	(0)444	</home phone prefix>
<home phone>	520045	</home phone>
<home fax>	75668	</home fax>
<work company>	Università di Padova	</work company>
<work dept>	Dip. Elettronica Informatica	</work dept>
<work title>	Programming Expert	</work title>
<work address>	via Gradenigo 6/A	</work address>
<work city>	Padova	</work city>
<work zip>	35100	</work zip>
<work state>	Padova	</work state>
<work country>	Italia	</work country>
<work country prefix>	++39	</work country prefix>
<work phone prefix>	(0)49	</work phone prefix>
<work phone>	456789	</work phone>
<work fax>	987654	</work fax>
<date>	31/12/1997	</date>
For CUSTOM TAGS :		
<custom :fieldname>	***fieldname is a user string	</custom :fieldname>

Section

9

Schemes of .FOR files rules

A format file (.FOR) is a text tagged (similar to HTML) file. There's the possibility to use 3 fields (we will use the word field for .FOR file and the word tag for Business Cards).

<TEXT> ... </TEXT>
<DATA> ... </DATA>
<IFANY> ... </IFANY>

Attention : there's **no** case sensitivity.

This is the skeleton of their use :

[<TEXT> ... </TEXT>]
<DATA>
[<IFANY> ... </IFANY>]
</DATA>
[<TEXT> ... </TEXT>]

Attention : **line feed** here are only for making it more readable.

The field **IFANY** is optional and can be inserted only inside a **DATA** field. No nesting is allowed for the other fields.

TEXT field:

What lies inside **<TEXT>** and **</TEXT>** is copied without any control in the PIM text file .PIM. It's not allowed to put other tags inside it because they will not be considered. This field it's useful to build text files that needs an header (e.g. Nowcontact) . TEXT is an optional field. Special characters '<' or '\ or '>' used inside the field TEXT must be written respectively as '\<' or '\\' or '\>'

This field is not obligatory, the same results may be obtained using the field DATA. It helps the user in writing more readable .FOR files.

DATA field:

Inside a field **DATA** is allowed to write:

- user text that will be copied "as it is" in .PIM
- tag of the Business Card
- field <IFANY> ... </IFANY> (optional)

Example:

<DATA>user text Business Card tags **<IFANY>** ... **</IFANY> </DATA>**

The user text inside **DATA** will be copied.
Nothing is written if a tag has no Business Card value associated, otherwise the tag is substituted by the values read in the Business Card.

This is the syntax for writing a tag :
<Tagname/Number/Separator>

- If no Number is specified the program will use the default Number (=1).

Example	**email** content in the Business Card	**output**
<email>	smith@dei.it smith@maya.it	smith@dei.it

- If Separator is specified and Number is 1 (or Number is not specified) the program will print a value followed by the separator if there's at least a corresponding value in the Business Card. It will print nothing if there's no corresponding value in the Business Card.

Example	**email** content in the Business Card	**output**
<email**/1/,**>	smith@dei.it smith@maya.it	smith@dei.it**,**
<email**/,**>	smith@dei.it smith@maya.it	smith@dei.it**,**
<email**/1/,**>		

- If no Separator is specified there will be no separators between the values inserted.

Example	**email** content in the Business Card	**output**
<email/2>	smith@dei.it smith@maya.it	smith@dei.itsmith@maya.it

- Separator can contain an arbitrary number of characters.

Example	**email** content in the Business Card	**output**
<email/2/**,- !**>	smith@dei.it smith@me.it	smith@dei.it**,- !**smith@me.it

- Number 0 indicates that all the Business Cards values associated to that tag must be copied down. In this situation if the number of Business Card values available is 0 or 1 no separator is printed, if the number of values is n, the program will print n-1 separators.

Example	**email** content in the Business Card	**output**
<email/**0/,**>	smith@xa.it smith@wb.it	smith@xa.it**,**smith@wb.it
<email/**0/,**>	smith@ze.it	smith@ze.it

- If Number is n>1 the separator will always be printed n-1 times even if a different number of values is present in the Business Card.

Example	**email** content in the Business Card	**output**
<email/**4/,**>	smith@xa.it smith@wb.it	smith@xa.it**,**smith@wb.it**,,**
<email/**4/,**>	smith@ze.it	smith@ze.it**,,,**
<email/**4/,**>		**,,,**

- There's also the possibility to define a separator writing its ASCII code in decimal format preceded by the character $. It's possible to use more than a separator character.

Example	**email** content in the Business Card	**output**
<email/2/**$44**>	smith@xa.it smith@wb.it	smith@xa.it**,**smith@wb.it
<email/2/**$44$32**>	smith@xa.it smith@wb.it	smith@xa.it**,** smith@wb.it
<email/2/**$44&$37**> >	smith@xa.it smith@wb.it	smith@xa.it**,&%**smith@wb.it

- For using a special character '<' or '>' or '/' or '$' or '\' in the entire tag or data field **every** special character must be preceded by a character '\'

Example	**email** content in the Business Card	**output**
<email/2/,**\>**>	smith@xa.it smith@wb.it	smith@xa.it,**>**smith@wb.it
<email/2/,**\>\$**>	smith@xa.it smith@wb.it	smith@xa.it,**>$**smith@wb.it
<email/2/,**\>\>**>	smith@xa.it smith@wb.it	smith@xa.it,**>>**smith@wb.it
Example	**custom my->$** content in the Business Card	**output**
<custom :my-\>$>	2,000	2,000

Attention : There's **no syntax error** advice. Not correct tag specification may cause unforeseeable behaviour of the program.

IFANY field :

The IFANY field must be always inside a DATA field in the sense that if it's present a field IFANY there must be the structure **<DATA>...<IFANY>...</IFANY>...</DATA>**

IFANY is an optional field. Inside the field <IFANY>...</IFANY> is possible to write text and tags of the Business Card but this text will be copied in the output file only if there's at least a value of the associated tags in the Business Card. In particular if inside a field IFANY there are more Business Cards tags the text will be written if at least a tag has a value associated.

Examples :

Example	**email** Business Card	**name** Business Card	output
<DATA> Info : **<IFANY>** this is an email : <email> and this is a name : <name> **</IFANY>** **</DATA>**	smith@ze.it	john	Info : this is an email : smith@ze.it and this is a name : john

Example	**email** Business Card	**name** Business Card	output
<DATA> Info : **<IFANY>** this is an email : <email> and this is a name : <name> **</IFANY>** **</DATA>**			Info :
<DATA> Info : **<IFANY>** this is an email : <email> and this is a name : <name> **</IFANY>** **</DATA>**		john	Info : this is an email : and this is a name : john

The IFANY field is essential for those PIMs that have no separators at the end of the individual data if there's an empty field and for having always the output under control whatever could be the fields content.

A good version for the above example is :

<DATA> Info :
<IFANY> this is an email : <email> **</IFANY>**
<IFANY> and this is a name : <name> **</IFANY>**
</DATA>

Section

9

More about HTML pages

Specification

The Business Cards tags are just used as HTML tags. That is there's a begin tag and an end tag.

<TagName> user text **</TagName>**

TagName is one of the Business Cards valid tag (also custom tags are allowed) (see p. 21 table of the standard tags). The user text inserted between 2 tags (<TagName> </TagName>) will be replaced by new text information, actually in the Business Card, related to TagName.

Attention : there's **no** case sensitivity

Section
11

The alias file "ALIAS.BCA"

Introduction

An **alias** tag fixes an **association** between standard and/or custom tags.

For example suppose that there is a sender with two home phones. He couldn't write in the business card two times the **<home phone>** tag.

To obviate to the problem the sender defined a custom tag like:

<custom:second home phone>

Then the receiver can define this tag as an <home phone> alias. That is he can create an alias file :

<usr :phone>=<home phone> <custom:second home phone> ;

When in the format file is specified an alias tag, all the values of the fields associated to that alias are copied in the output file.

Using alias has these advantages:

- allows to associate different tags having the same meaning (for example useful with custom tags written in different language)
- supply a way to obviate the business card tags fixed structure.

Alias Files Specification

The alias used by the program must be defined in a text file named **“alias.bca”** put in the program directory.

This file is optional, if it's not present the program produces a warning message and it doesn't use alias, but proceeds correctly.

Every Alias File containing alias table should respect the following syntax.

<usr :alias1> = <tag1> <tag2> <tag3> ... <tagn> ;
<usr :alias2> = <tag1> <tag2> <tag3> <tag4>... <tagm> ;

...
Where tag1 can be a standard tag (e.g. name) or a custom tag (e.g. custom :nickname)

Attention : **separators** between tag names can be: none, spaces or tabulation characters.

It is an **error** to write:

<usr:identifier>=;

<usr:identifier>=

If we have such an "alias.bca" file:

<usr:phone>=<home phone> <custom:second home phone> ;

in the format file we can write:

<DATA>**<usr:phone/2/,>**<DATA>

Example with lotus organizer format:

- File alias:

  <usr:address>=<home address> <work address> <custom:sea house address> <custom:mountain house address>;

- Values in the business card database:

home address	via Treviso 31
work address	via Gradenigo 6/A
sea house address	via Giotto 12
mountain house address	via Vicenza 7

- Format file (no line feed):

  <DATA>"<surname/$32><name>","**<usr:address/4/",">**","<home zip>","<home city>","<home state>","<custom:login>"</DATA>

- Output: file

..... ,"via Treviso 31","via Gradenigo 6/A","via Giotto 12","via Vicenza 7",.....